Security Programmable Metamaterial Antenna Array for Physical Layer Security
Open House

Prachi Patel and Dylan Turner

Introductions:

Dylan Turner

Prachi Patel

Motivation

- Increasing need to develop **low cost** security protocol.
- Any reduction in cost affects **tens of billions of devices**.
- Investigate a **computationally simpler** security solution.

The Internet of Things (IoT)
Collective Network of Billions of Devices

Solution

???

???

Security Programmable

Metamaterial Antenna Array

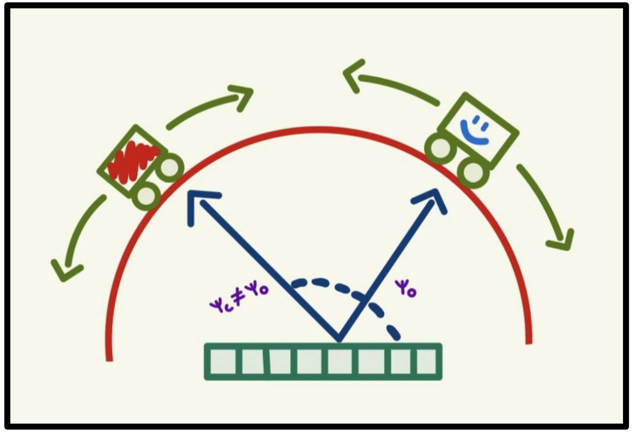
for Physical Layer Security

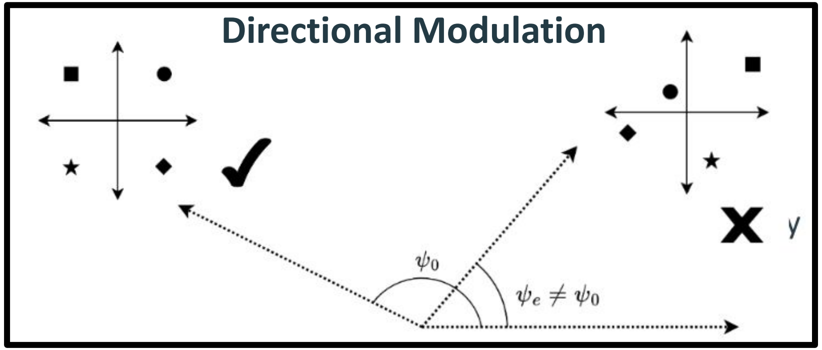
???

???

Physical Layer Security

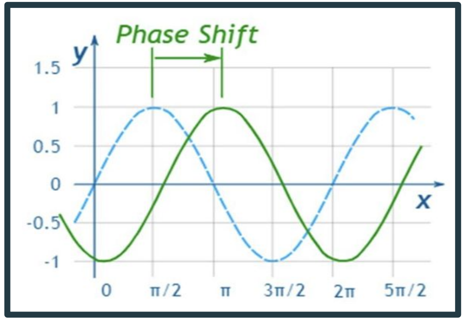
- Exploiting **physical properties** of a system to provide security against eavesdropper.
- This project focuses on **Directional Modulation**
- How do we achieve this?





Metamaterial Antenna Array (MTM)

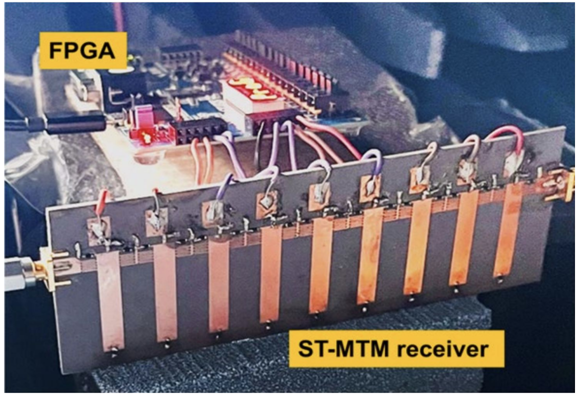
- A material which exhibits properties that are **not found in nature**.
  - Eg: Applies a variable phase shift to a signal dependent on voltage.



Applied voltage: small shift

No voltage: large shift

- Several Metamaterial Antennas can be put together to form an array.

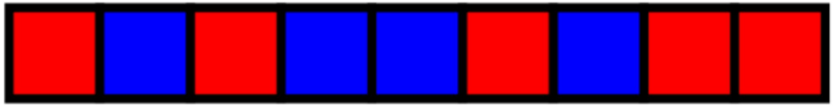


Security Programmable

- **Programmable via codes** which are applied to the Metamaterial array.

- Phase shifts from the elements will cause the signal to **scramble itself in undesired directions**.

Injected signal passes through each Metamaterial Element



Red = Voltage on | Blue = Voltage off

Security Programmable

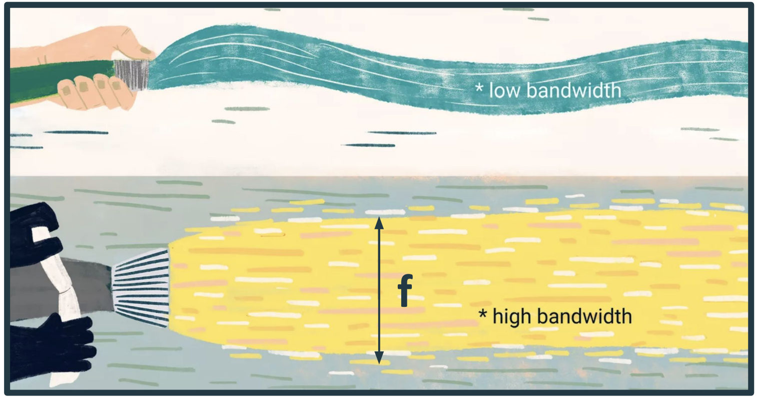
Metamaterial Antenna Array

for Physical Layer Security

The Problem

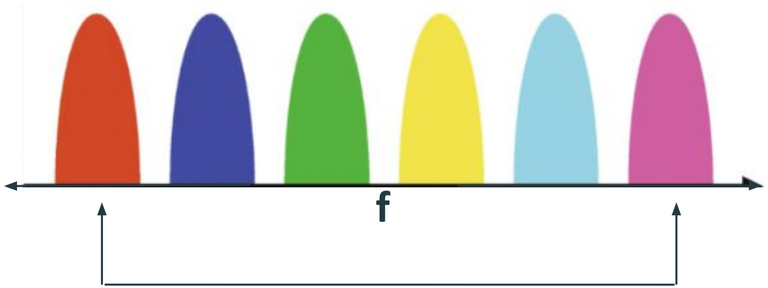
Problem with Sending More Data

Faster Data Rate → Wider bandwidth



Signal must be spread across **wider band of frequencies**

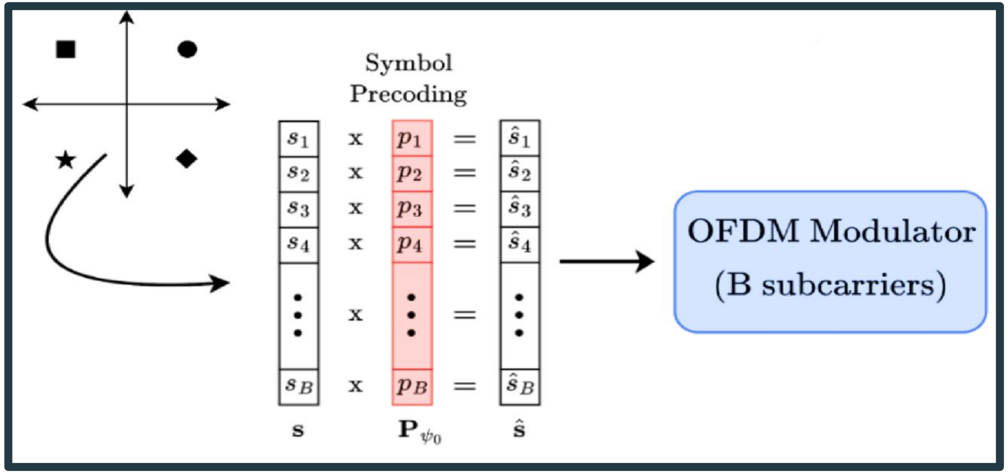
Problem: Antenna is Frequency Dependent



“Parts of the signal” with **large differences in frequency** will be affected **differently.**

Symbol Precoding

Solution: Multiply the information by a “Precoding Vector” that negates the effects of the antenna.



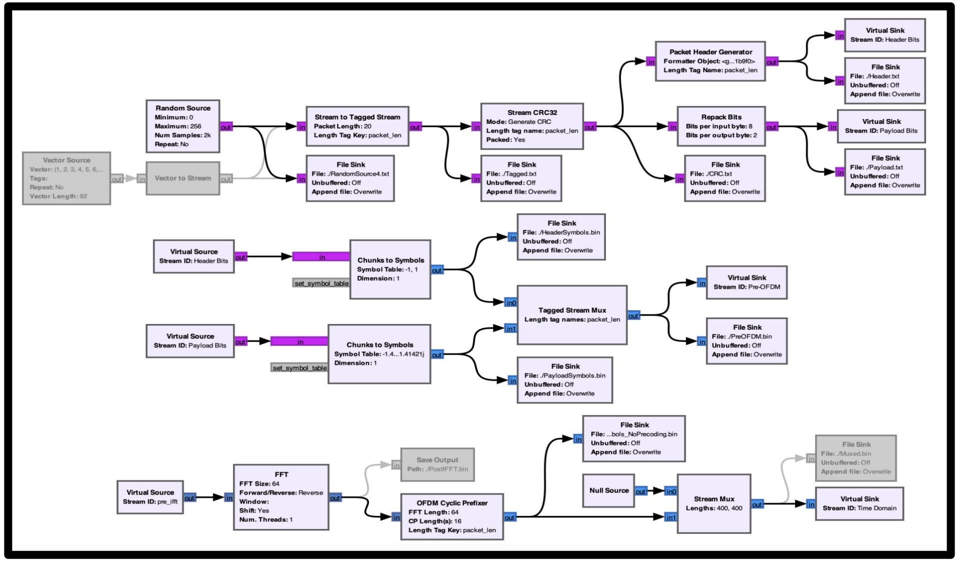
$$w(\nu, L, \psi, b) = \frac{\operatorname{sinc}\left(\frac{\nu\pi}{L}\right)}{L} e^{j\pi\nu/L} \left\{ \sum_{u=1}^{L} M_{u,b}(\psi, \mathbf{q}_u)\ e^{-j2\pi\nu u/L} \right\}$$

$$M_{u,b}(\psi, \mathbf{q}_u) = \sum_{n=1}^{N} \Gamma_{n,b}(\mathbf{q}_u) e^{-\alpha(n-1)p} e^{j(n-1)k_0 p \cos(\psi)}$$

$$\Gamma_{n,b}(\mathbf{q}_u) = \prod_{i=1}^{n} e^{j\kappa_{\mathbf{q}_u(i)}}, \kappa_{\mathbf{q}_u(i)} = \begin{cases} \beta_0(b)p & \text{if } \mathbf{q}_u(i) = 0 \\ \beta_1(b)p & \text{if } \mathbf{q}_u(i) = 1 \end{cases}$$

GNU Radio

Open Source free toolkit for creating signal processing systems through software.



GNU Radio Implementation of our Transmitter

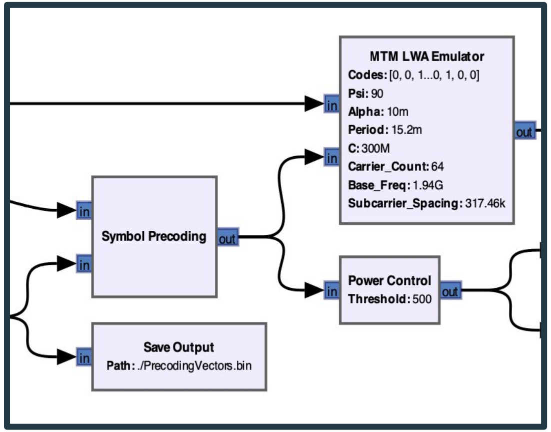
```python
def calc_w(self, nu, L, codes, alpha, period, b, psi, c):
    #Loop over L
    total_M = 0
    for u in range(1, codes.shape[0] + 1):
        total_M += self.calc_m(codes[u-1], alpha, period, b, psi, c) * np.exp(-1j
    return total_M * (np.sinc(nu / L) / L) * np.exp(1j * nu * np.pi / L)

def calc_m(self, code_u, alpha, period, b, psi, c):
    #Loop over N
    psi = 180 - psi
    total_gamma = 0
    for n in range(1, code_u.shape[0]+1):
        total_gamma += self.calc_gamma(n, b, code_u) * (np.exp(-alpha * (n-1) * pe
    return total_gamma

def calc_gamma(self, n, b, code_u):
    #Beta Values for bth carrier
    beta_0 = (1.488e-7 * (b * self.subcarrier_spacing + self.base_freq) - 338.46)
    beta_1 = (1.164e-7 * (b * self.subcarrier_spacing + self.base_freq) - 216.63)
```

Snippet of Symbol Precoding Python Code

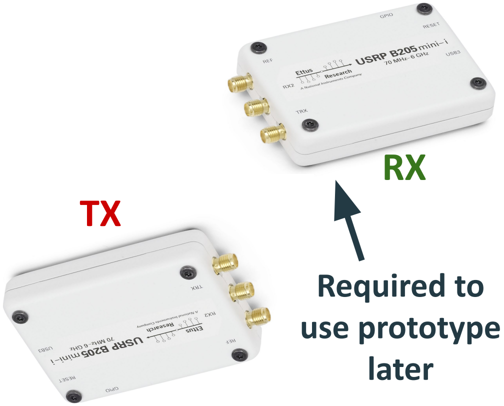
Testing

Testing Phases

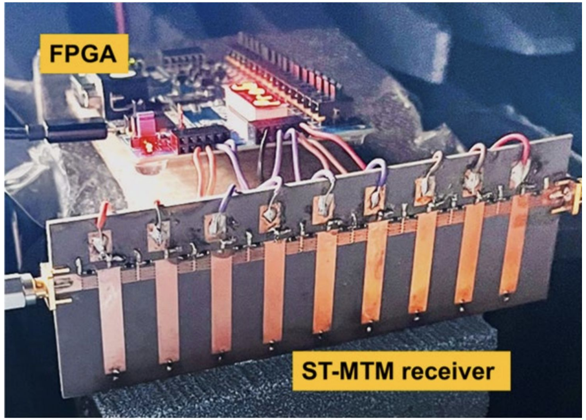
1. Simulation



2. Basic Equipment



3. MTM Antenna Array

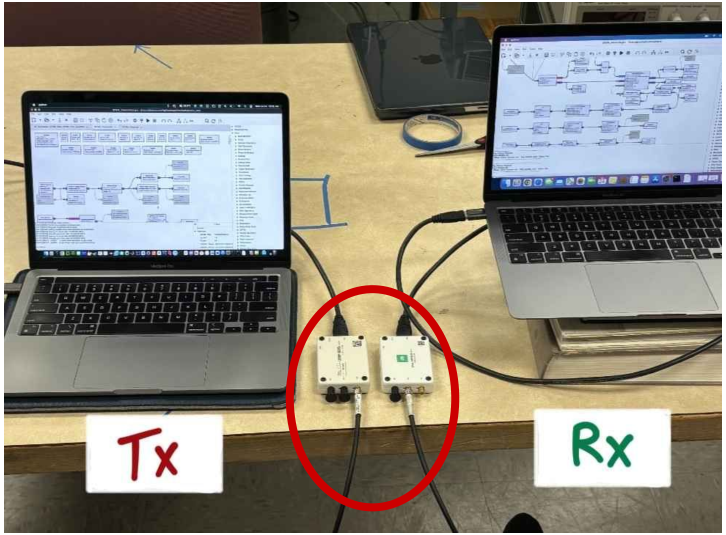


Goal: ensure systems works at each stage before introducing more unknown variables.

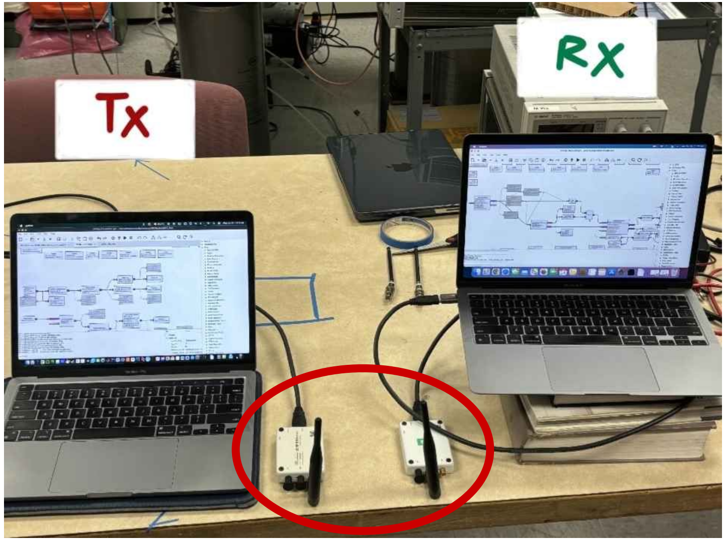
Basic Equipment Testing

Goal: Transmit and receive a wideband signal using the B205 Mini-i USRP

<u>Wired Connection</u>



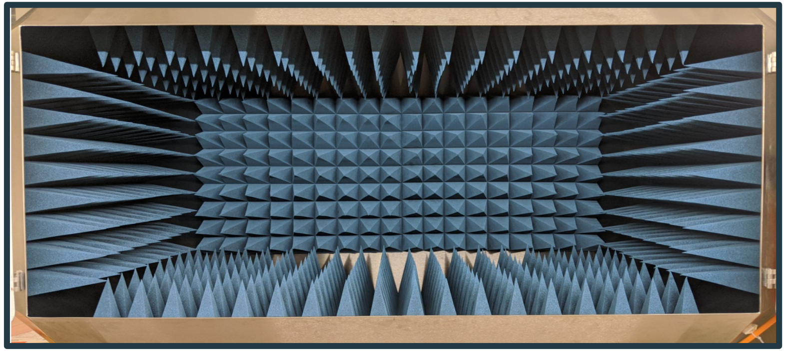
<u>Antenna</u>

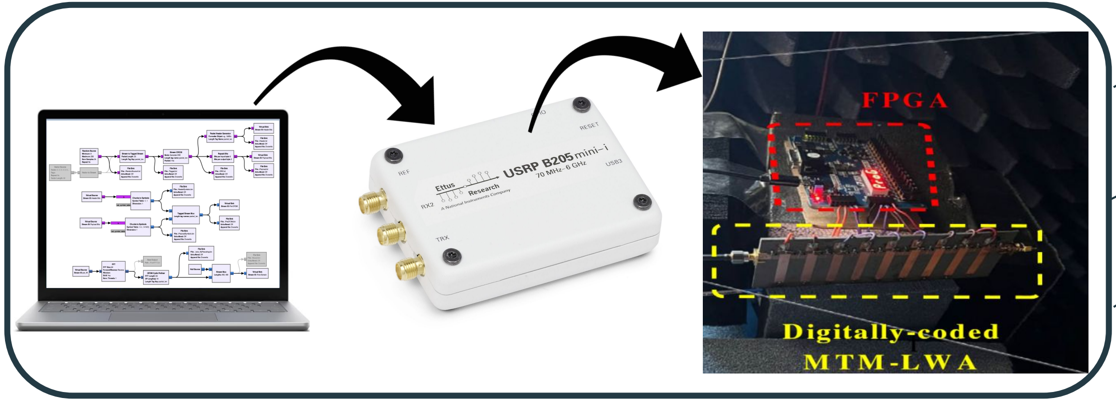


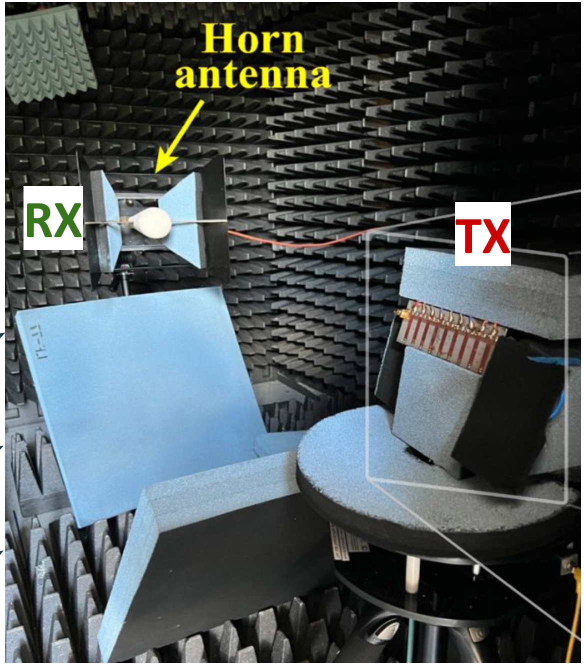
These USRPs will be needed later to connect to our Antenna Prototype

Anechoic Chamber

Allows for measurement of a systems performance without external interference or reflections



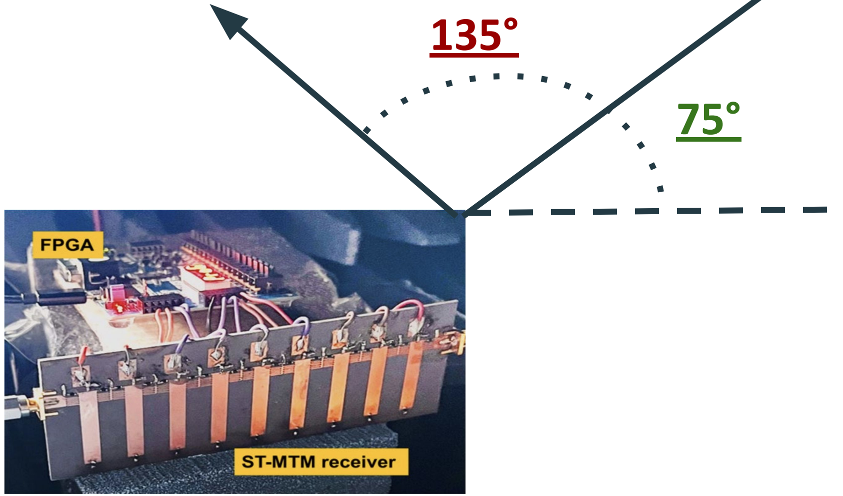




Results

Scrambled Signal

```
header_payload_demux :info: Parser returned #f
packet_headerparser_b :info: Detected an invalid packet at item 7872
header_payload_demux :info: Parser returned #f
packet_headerparser_b :info: Detected an invalid packet at item 7920
header_payload_demux :info: Parser returned #f
packet_headerparser_b :info: Detected an invalid packet at item 7968
header_payload_demux :info: Parser returned #f
packet_headerparser_b :info: Detected an invalid packet at item 8016
header_payload_demux :info: Parser returned #f
packet_headerparser_b :info: Detected an invalid packet at item 8064
header_payload_demux :info: Parser returned #f
```



Successful Transmission

```
----------------------------------------------------------------------
Tag Debug: Payload
Input Stream: 00
 Offset: 0  Source: n/a    Key: frame_len   Value: 34
 Offset: 0  Source: n/a    Key: packet_num   Value: 1157
 Offset: 0  Source: n/a    Key: packet_len   Value: 1588
 Offset: 0  Source: n/a    Key: ofdm_sync_chan_taps   Value: #[(0,-0) (0,-0) (0,-0) (0,-0) (0,-0) (0,-0) (4.95103e-05,0.000554926) (-0.0032586,0.000181324)
(5.88183e-05,0.00232299) (0.000751956,0.00120249) (-0.00254272,0.000306642) (-0.00192348,-0.00528128) (0.00132008,0.000850759) (-0.0010975,0.00276146)
(-8.03692e-05,0.00212614) (0.000740792,-0.00431434) (-0.00268088,0.00239754) (0.00142883,-0.000137892) (0.00286203,-0.00121771)
(0.00271821,8.9186e-05) (-0.00143348,-0.00368198) (0.00288561,-9.53536e-05) (-0.00278513,0.00126042) (-0.00194153,0.0025178) (-0.0030899,0.00400383)
(-0.00987739,0.00785388) (-0.00810666,-0.000430975) (0.00903368,-0.000883272) (-0.0107201,0.00690506) (-0.00708761,0.0246502) (0.0219408,0.0129858)
(-0.0115142,-0.0861283) (0,-0) (0.00471223,0.0910247) (-0.00671487,0.00602554) (-0.00473997,-0.00102596) (-0.00595539,-0.00368735)
(-0.00422677,0.00186947) (0.00184556,0.0023007) (-0.00465478,-0.003495) (-0.0018473,0.000759621) (-0.00281591,-0.00269134) (0.0016484,0.00241287)
(-0.00098758,-0.00170475) (-0.00107155,-0.00133854) (0.000425142,-0.00305391) (-0.000356219,-0.000983322) (0.00229089,0.00217253)
(-0.00451028,-0.000837112) (0.00150928,-0.000167536) (0.00255357,9.9543e-05) (-0.0024076,0.00438214) (-0.00154195,-0.00132719)
(-0.000409748,0.00158569) (-0.0038427,-0.00345251) (-0.00207393,-0.000597466) (-0.00212224,0.00296569) (0.00336829,-0.00287269)
(0.00306859,-0.000787914) (0,-0) (0,-0) (0,-0) (0,-0) (0,-0)]
 Offset: 0  Source: n/a    Key: ofdm_sync_carr_offset   Value: 2
----------------------------------------------------------------------
```

Symbol Precoding works as a proof of concept!

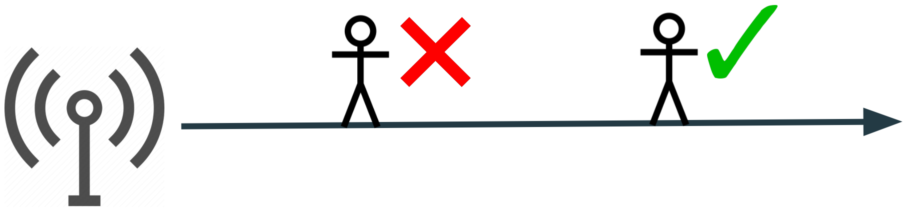
Future Work

Short Term

- **Automate measurements** with Python scripts.
  - Manually testing is imprecise and slow.
- **Determine power restrictions** to protect hardware.
  - To be safe, we had to be lenient before.

Long Term

- **Refine system** to ensure consistent reception of ALL packets.
- Incorporate additional layer of security:
  - **Account for distance**, not just angle.




THANK YOU